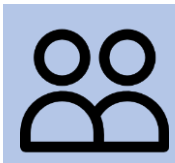
Taller en Sala Nro. 7
Listas Enlazadas

En la vida real, las listas enlazadas se usan para representar objetos en videojuegos http://bit.ly/2mcGa5w y para modelar pistas en juegos de rol http://bit.ly/2lPyXGC

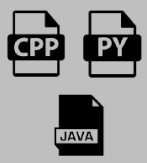
Las listas doblemente enlazadas se utilizan para representar procesadores de palabras como Microsoft Word, de esa forma es eficiente poder borrar una línea o insertar una línea en cualquier lugar del documento http://bit.ly/2foadEK



Trabajo en Parejas

Hoy, plazo máximo de entrega

Docente entrega código suelto en GitHub



Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código suelto por GitHub

Ejercicios a resolver

1. Implemente en el código los métodos *size* e *insert* de una lista simplemente enlazada.
2. Al código anterior agregue los siguientes métodos *remove* y *contains*

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

3. En la clase *Taller 7*, defina un método que calcule el máximo valor de los elementos de una lista enlazada de forma recursiva.

4. Implemente un algoritmo que permita comparar el contenido de dos listas. Este debe retorna true en caso de que ambas listas sean iguales o false en caso contrario.

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Ayudas para resolver los Ejercicios

Ayudas para el Ejercicio 1

Pista 1: Diferencien LinkedListMauricio *de LinkedList* del API de Java. Un error común es creer que todo se soluciona llamando los métodos existentes en *LinkedList* y, no es así, la idea es implementar una lista con arreglos nosotros mismos. A continuación, un ejemplo del error:

```
// Retorna el tamaño actual de la lista
public int size()
{
    return size();
}

// Retorna el elemento en la posición index
public int get(int index)
{
    return get(index);
}

// Inserta un dato en la posición index
public void insert(int data, int index)
{
    if (index <= size())
    {
        insert(data,index);
    }
}

// Borra el dato en la posición index
public void remove(int index)
{
    remove(index);
}

// Verifica si está un dato en la lista
public boolean contains(int data)
{
    return contains(data);
}
```

Pista 2: Tenemos la siguiente implementación de lista:

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
// Usar esto cuando se salga el índice
import java.lang.IndexOutOfBoundsException;
// Una lista simplemente enlazada
public class LinkedListMauricio {
   // Un nodo para una lista simplemente enlazada
    public class Node {
         public int data;
         public Node next;
         public Node(int data) {
          next = null;
    this.data = data;
         }
     }

private Node first;
private int size;
public LinkedListMauricio()
{
 size = 0;
 first = null;
}

 /**
  * Returns the node at the specified position in this list.
  * @param index - index of the node to return
  * @return the node at the specified position in this list
  * @throws IndexOutOfBoundsException
  */
 private Node getNode(int index) throws
IndexOutOfBoundsException {
    if (index >= 0 && index < size) {
       Node temp = first;
       for (int i = 0; i < index; i++) {
          temp = temp.next;
       }
       return temp;
    } else {
       throw new IndexOutOfBoundsException();
    }
 }

 /**
```

```
  * Returns the element at the specified position in this
list.
  * @param index - index of the element to return
  * @return the element at the specified position in this
list
  */
 public int get(int index) {
    Node temp = null;
    try {
       temp = getNode(index);
    } catch (IndexOutOfBoundsException e) {
       e.printStackTrace();
       System.exit(0);
    }

    return temp.data;
 }

// Retorna el tamaño actual de la lista
public int size()
{
 ...
}

// Inserta un dato en la posición index
public void insert(int data, int index)
{
 ...
}

// Borra el dato en la posición index
public void remove(int index)
{
 ...
}
// Verifica si está un dato en la lista
public boolean contains(int data)
{
 ...
}

}
```

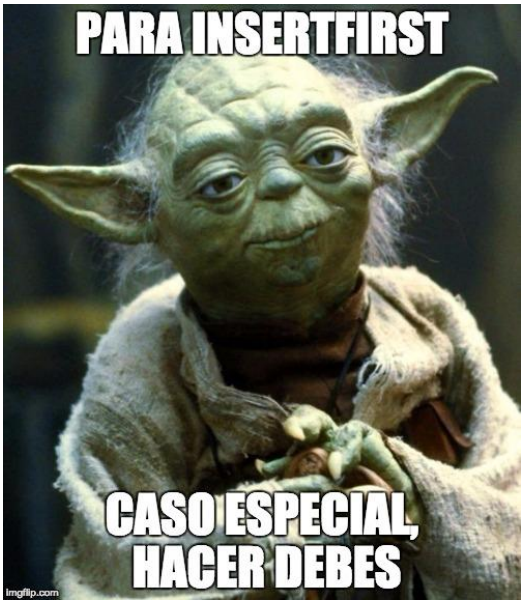
Como un ejemplo, para la lista [1,2,3,4], el método insert(10, lista.size() -1) agrega al final el número 10, quedando la lista [1,2,3,4,10]. El insert(3, 0) agrega al principio el número 3, quedando la lista [3, 1, 2, 3, 4, 10].

Error Común:

Ayudas para el Ejercicio 2

Error Común 1: Un error común, al hacer contains, es hacer un ciclo que llame n veces al método get porque como get es O(n), al llamarlo n veces queda el método contains $O(n^2)$ y se puede hacer en O(n). OJO.

Errores Comunes 2:



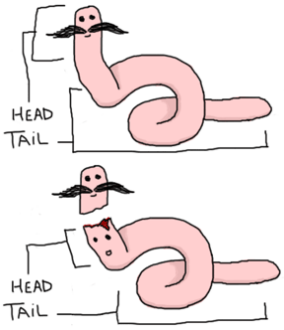
Ayudas para el Ejercicio 3

Pista 1: **Utilice su propia implementación de listas**, pues en la que trae Java en su API, no es posible acceder directamente a la clase nodo, y necesitaría usar iteradores en su lugar.

Pista 2: Paso 1: condición de parada

Pista 3:

```
private static int maximoAux(Nodo nodo) {
```




```
}
public static int maximo(LinkedListMauricio lista) {
  return maximoAux(lista.first);
}
```

Como un ejemplo si tenemos una lista = [1,2,3,4], el método maximo(lista) retorna 4.

Error Común:

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Ayudas para el Ejercicio 4

Como un ejemplo comparar([1,2,3], [1,2,3]) retorna verdadero y comparar([1,2,3],[1,2,3,4]) retorna falso.

Error Común:

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 **Ext.** 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agende una cita con él a través de **http://bit.ly/2gzVg10** , en la pestaña *Semana. Si no da clic en esta pestaña, parecerá que toda la agenda estará ocupada.*